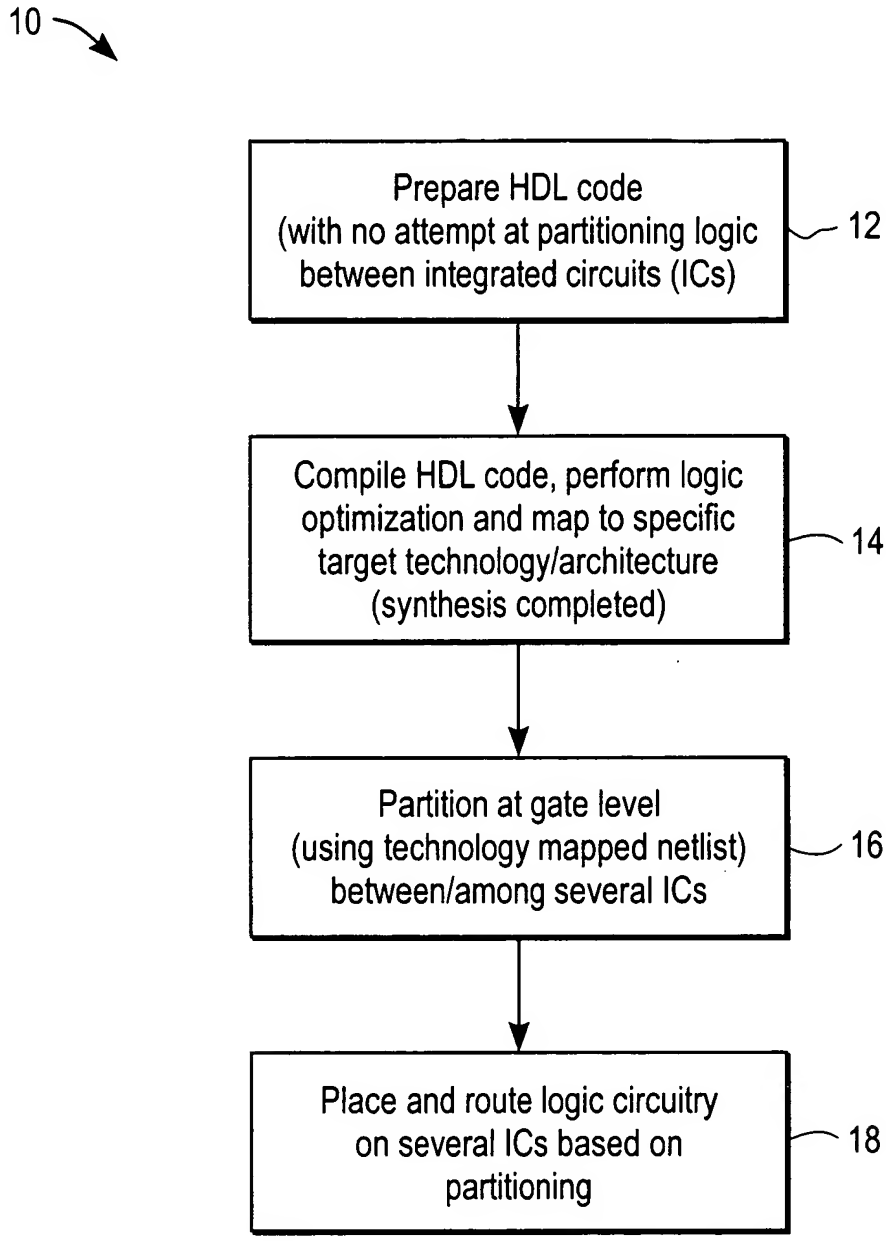

FIG. 1A (PRIOR ART)

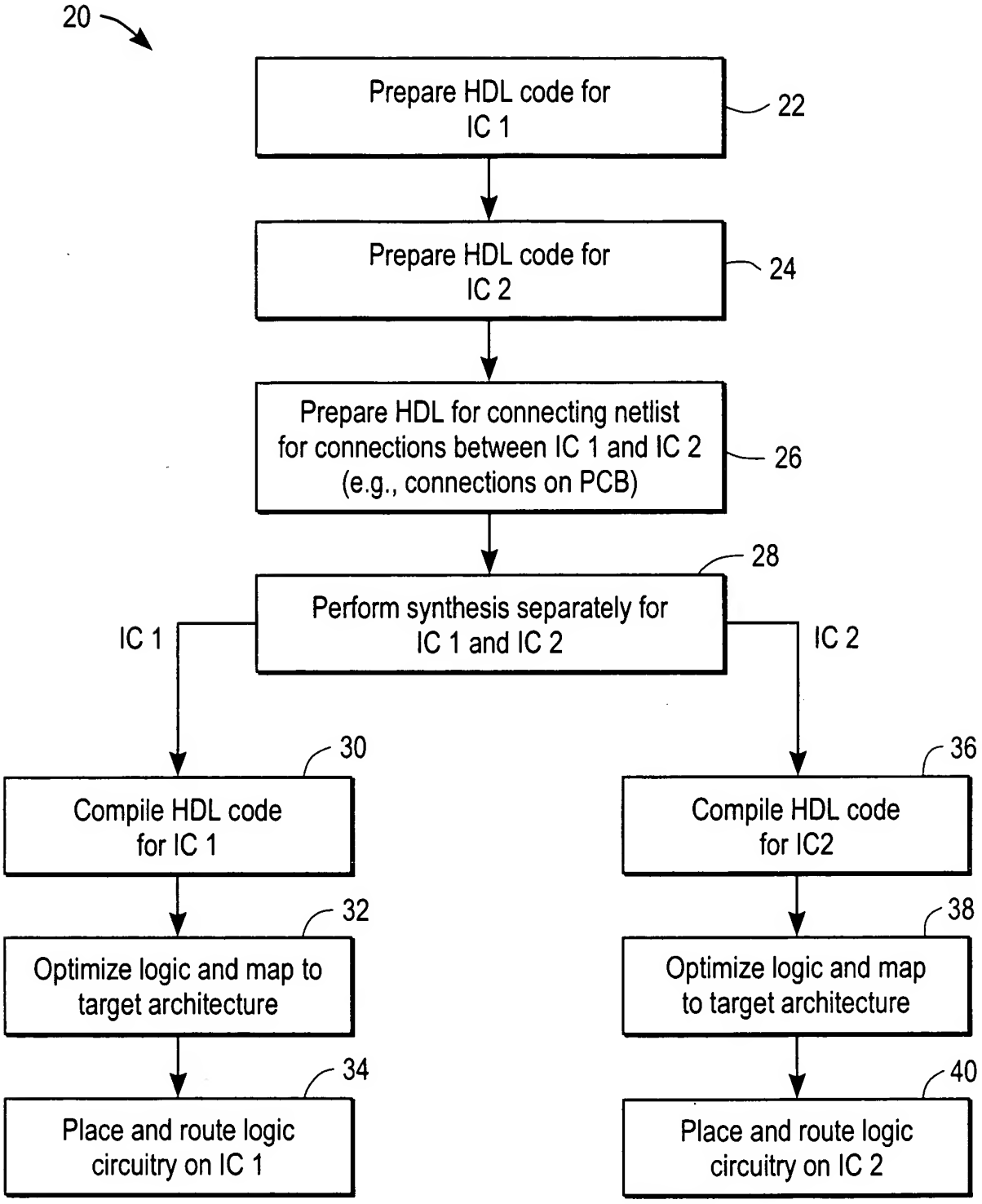

FIG. 1B (PRIOR ART)

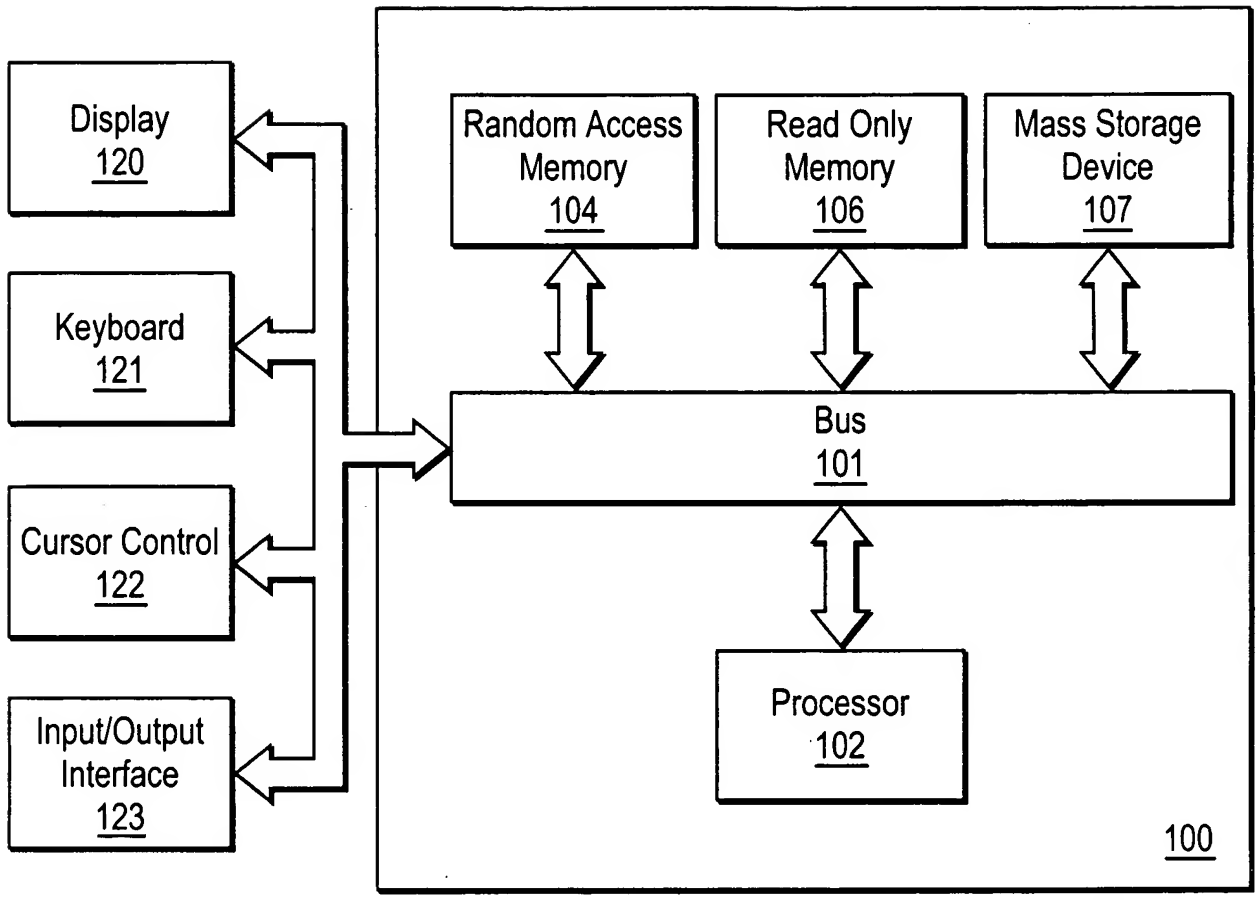
Display
120
Keyboard
121
Cursor Control
122
Input/Output
Interface
123
Random Access
Memory
104
Read Only
Memory
106
Mass Storage
Device
107
Bus
101
Processor
102
100

FIG. 2

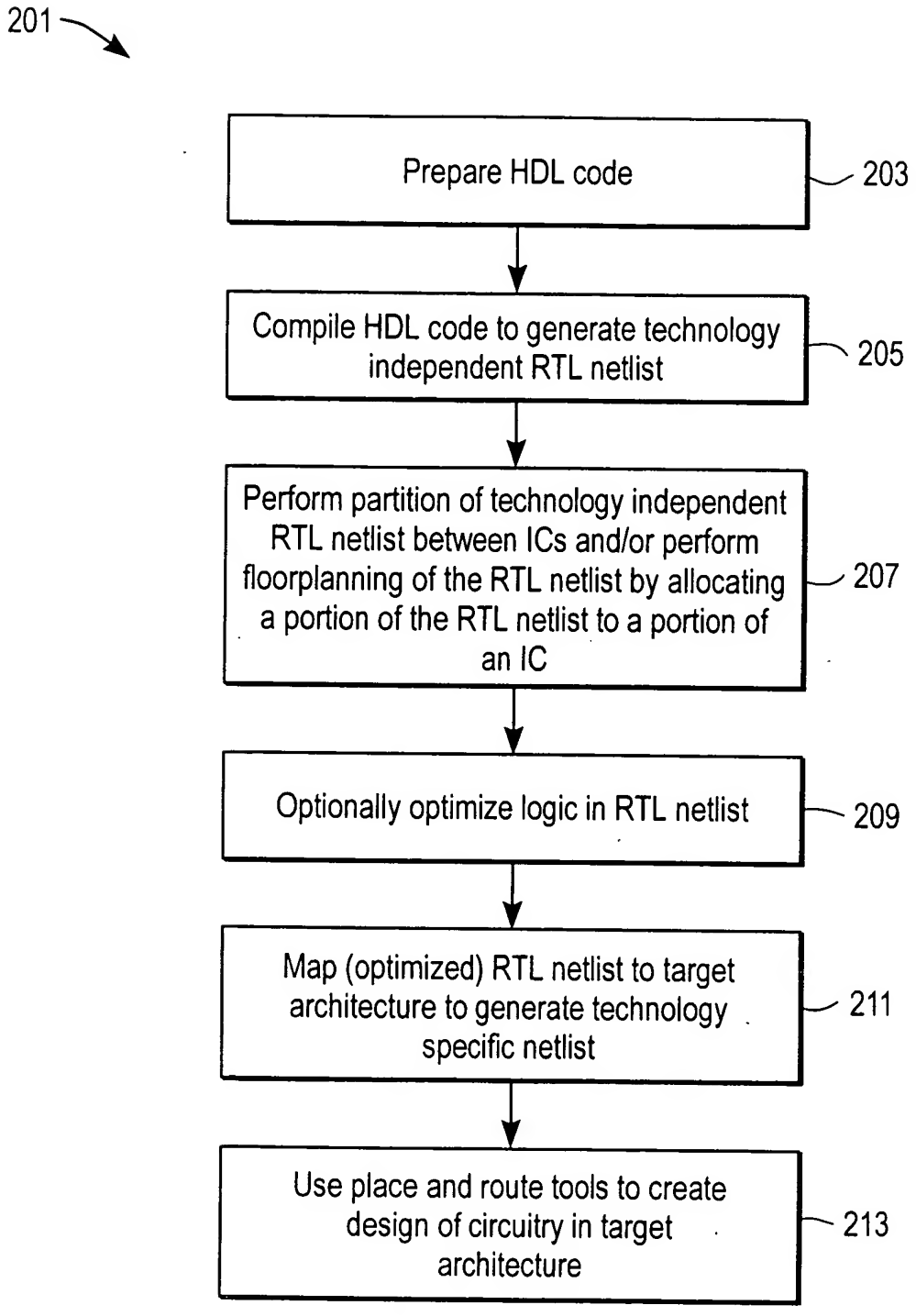
201
Prepare HDL code
203
Compile HDL code to generate technology independent RTL netlist
205
Perform partition of technology independent RTL netlist between ICs and/or perform floorplanning of the RTL netlist by allocating a portion of the RTL netlist to a portion of an IC
207
Optionally optimize logic in RTL netlist
209
Map (optimized) RTL netlist to target architecture to generate technology specific netlist
211
Use place and route tools to create design of circuitry in target architecture
213

FIG. 3

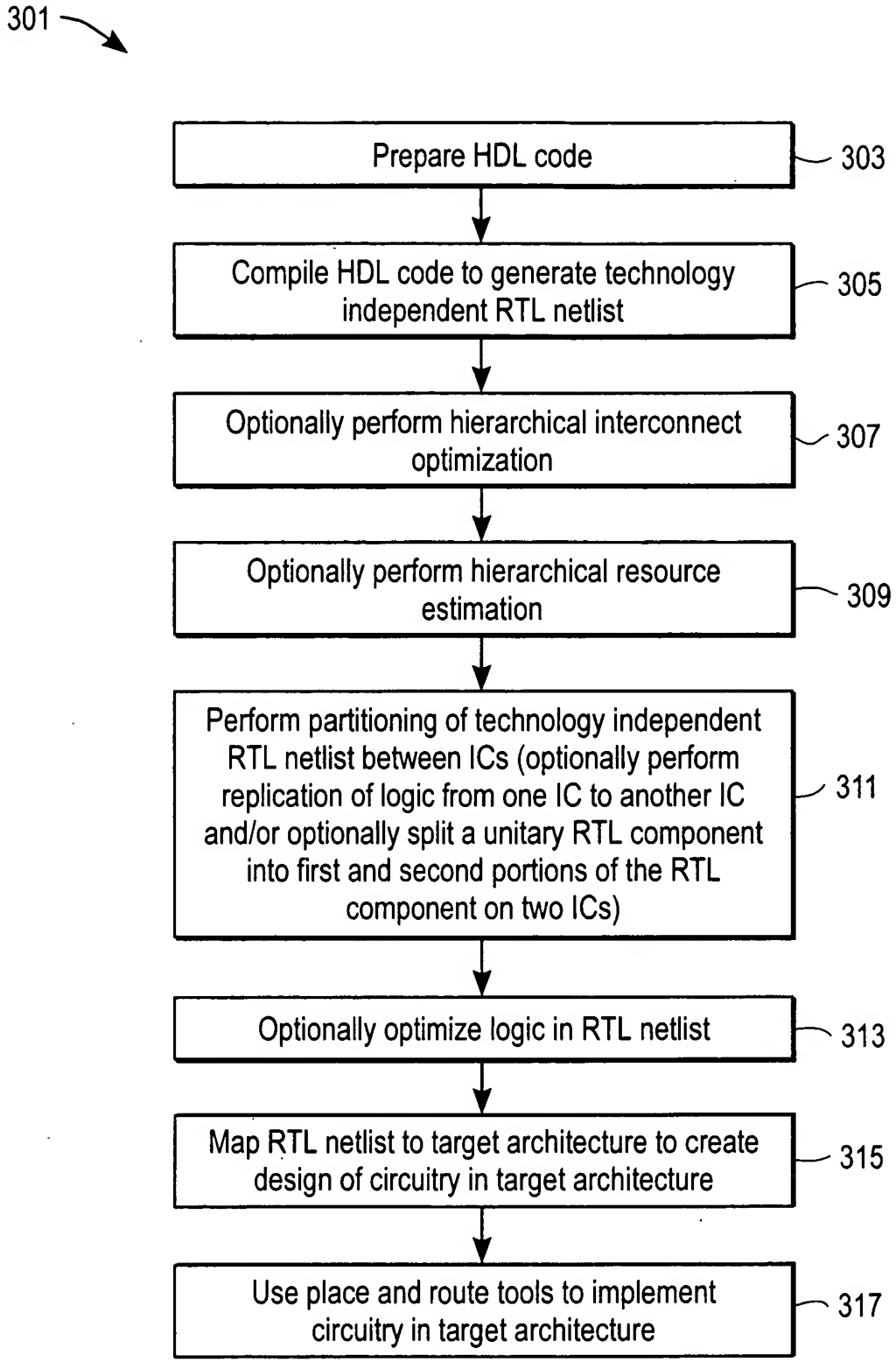
301
Prepare HDL code
303
Compile HDL code to generate technology independent RTL netlist
305
Optionally perform hierarchical interconnect optimization
307
Optionally perform hierarchical resource estimation
309
Perform partitioning of technology independent RTL netlist between ICs (optionally perform replication of logic from one IC to another IC and/or optionally split a unitary RTL component into first and second portions of the RTL component on two ICs)
311
Optionally optimize logic in RTL netlist
313
Map RTL netlist to target architecture to create design of circuitry in target architecture
315
Use place and route tools to implement circuitry in target architecture
317

FIG. 4A

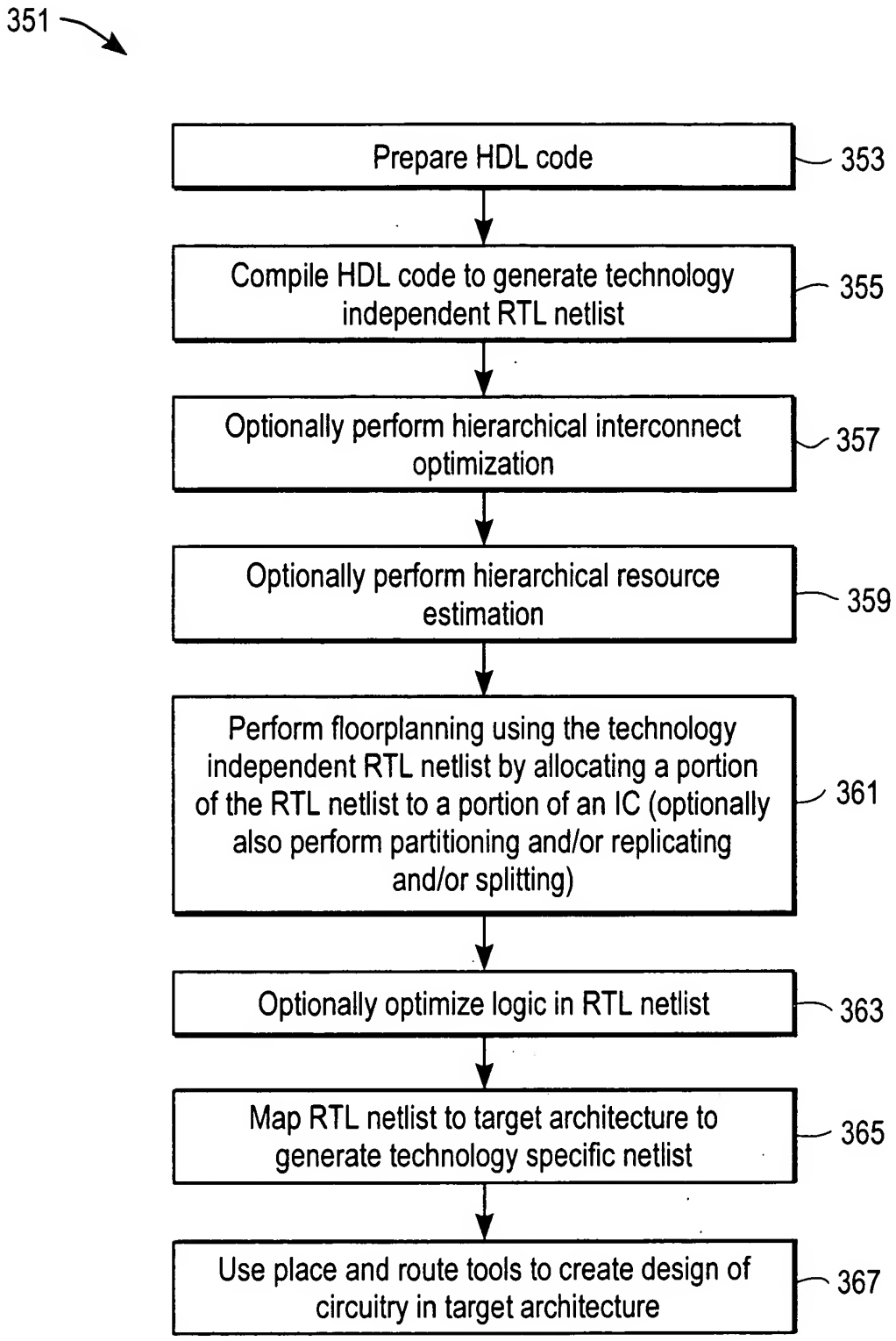
351
Prepare HDL code
353
Compile HDL code to generate technology independent RTL netlist
355
Optionally perform hierarchical interconnect optimization
357
Optionally perform hierarchical resource estimation
359
Perform floorplanning using the technology independent RTL netlist by allocating a portion of the RTL netlist to a portion of an IC (optionally also perform partitioning and/or replicating and/or splitting)
361
Optionally optimize logic in RTL netlist
363
Map RTL netlist to target architecture to generate technology specific netlist
365
Use place and route tools to create design of circuitry in target architecture
367

FIG. 4B

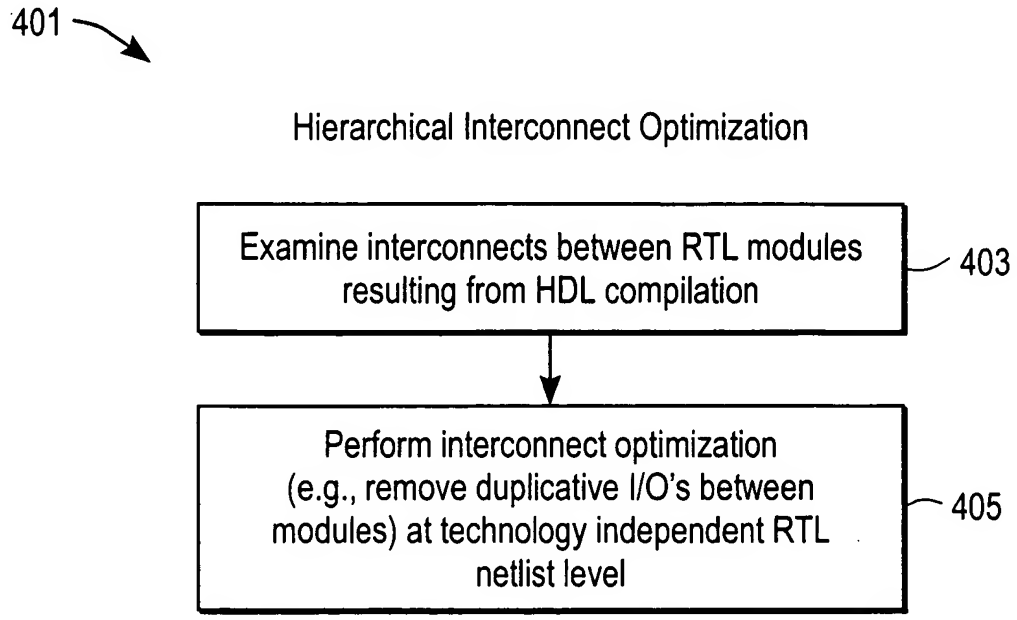
401
Hierarchical Interconnect Optimization
Examine interconnects between RTL modules resulting from HDL compilation
403
Perform interconnect optimization (e.g., remove duplicative I/O's between modules) at technology independent RTL netlist level
405

FIG. 5A

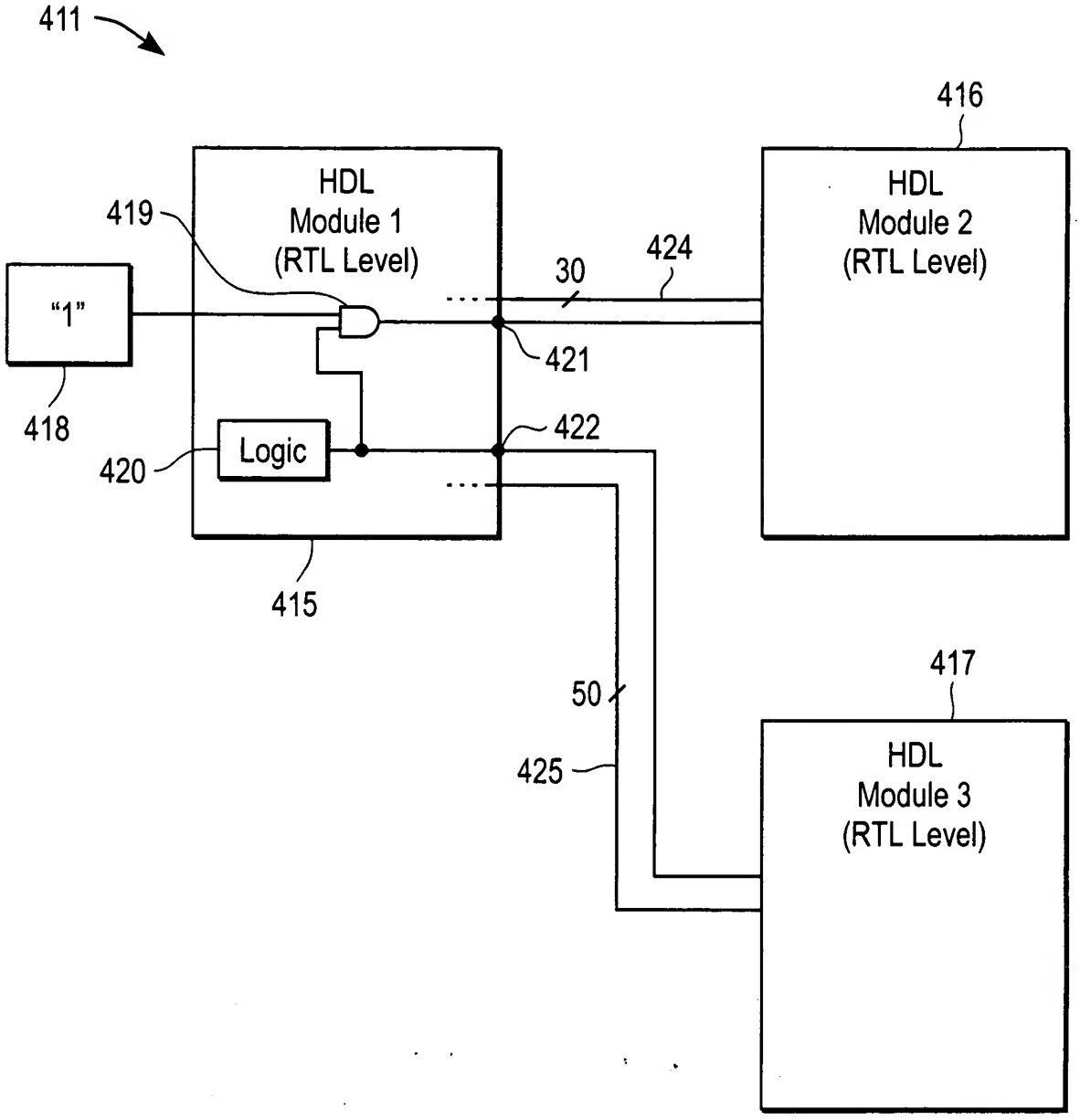
411
416
HDL
Module 1
(RTL Level)
HDL
Module 2
(RTL Level)
419
424
30
"1"
421
418
422
Logic
420
415
417
50
425
HDL
Module 3
(RTL Level)

FIG. 5B

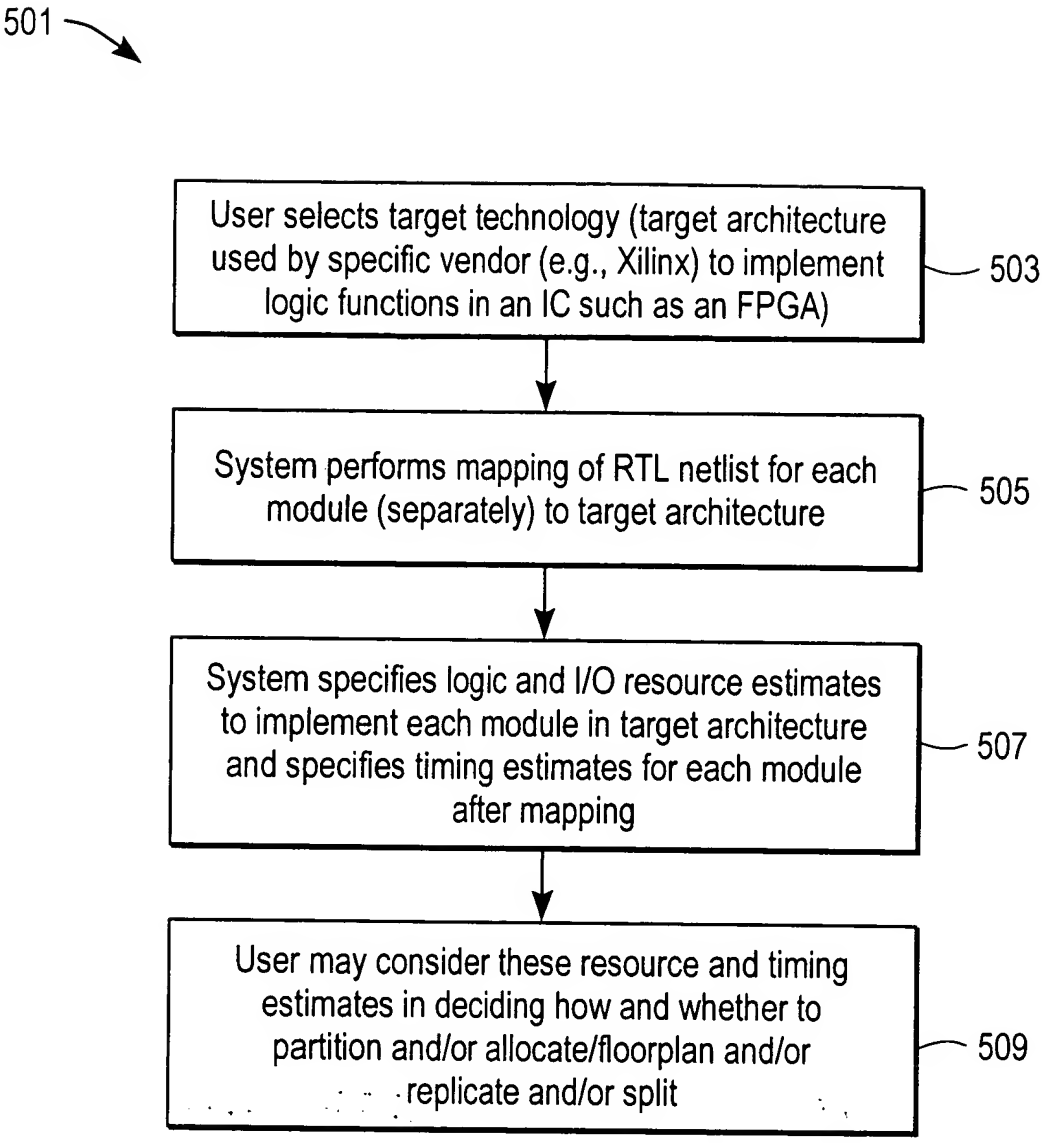
501
User selects target technology (target architecture used by specific vendor (e.g., Xilinx) to implement logic functions in an IC such as an FPGA)
503
System performs mapping of RTL netlist for each module (separately) to target architecture
505
System specifies logic and I/O resource estimates to implement each module in target architecture and specifies timing estimates for each module after mapping
507
User may consider these resource and timing estimates in deciding how and whether to partition and/or allocate/floorplan and/or replicate and/or split
509

FIG. 6

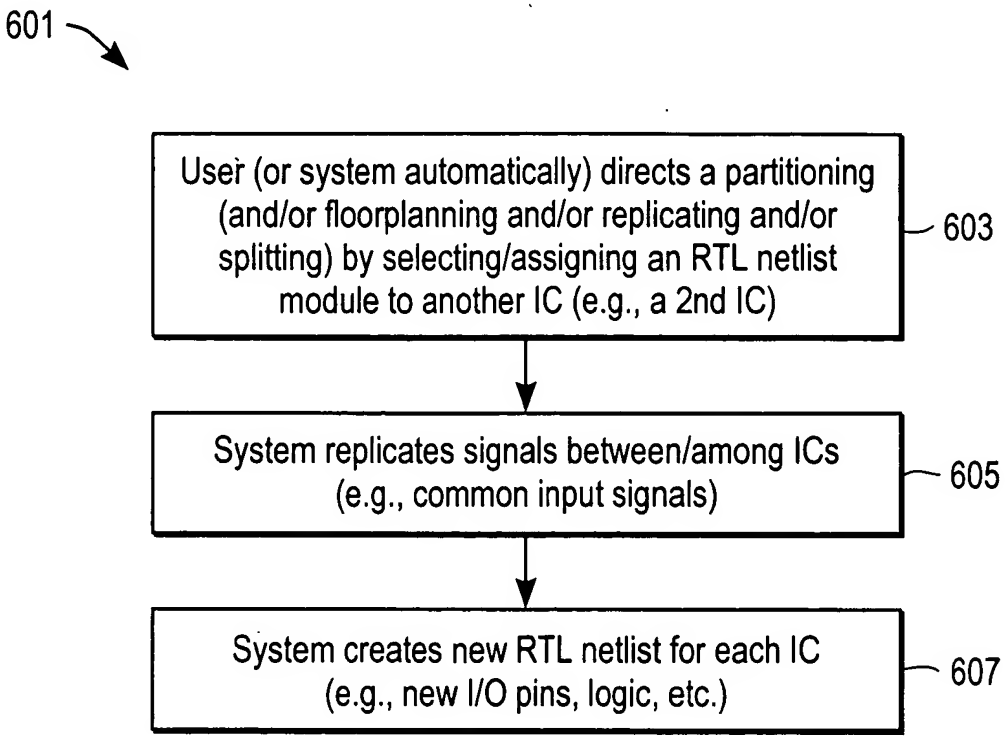
601
User (or system automatically) directs a partitioning (and/or floorplanning and/or replicating and/or splitting) by selecting/assigning an RTL netlist module to another IC (e.g., a 2nd IC)
603
System replicates signals between/among ICs (e.g., common input signals)
605
System creates new RTL netlist for each IC (e.g., new I/O pins, logic, etc.)
607

FIG. 7A

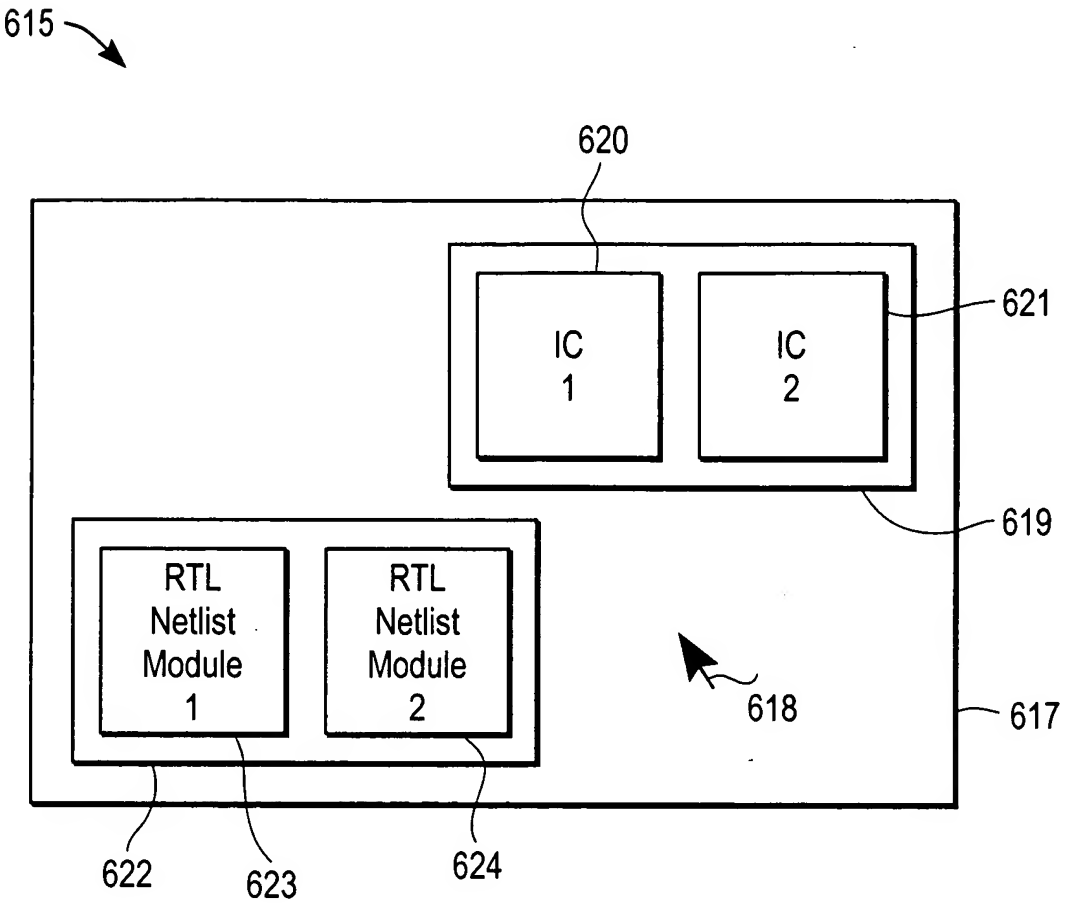
615
620
IC
1
IC
2
621
619
RTL
Netlist
Module
1
RTL
Netlist
Module
2
618
617
622
623
624

FIG. 7B

```
module prep2_2 (DATA0, DATA1, DATA2, LDPRE, SEL, RST, CLK,LDCOMP);
output [7:0] DATA0;
input [7:0] DATA1, DATA2;
input LDPRE, SEL, RST, CLK, LDCOMP;
        wire [7:0] DATA0_internal;
        prep2_1 inst1 (CLK, RST, SEL, LDCOMP, LDPRE, DATA1, DATA2, DATA0_internal);
        prep2_1 inst2 (CLK, RST, SEL, LDCOMP, LDPRE, DATA0_internal, DATA2, DATA0);

endmodule
```

703

```
module prep2_1 (CLK, RST, SEL, LDCOMP, LDPRE, DATA1, DATA2, DATA0);
input CLK, RST, SEL, LDCOMP, LDPRE;
input [7:0] DATA1, DATA2;
output [7:0] DATA0;
reg [7:0] DATA0;
reg [7:0] highreg_output, lowreg_output; // internal registers

wire compare_output = (DATA == lowreg_output); // comparator
wire [7:0] mux_output = SEL ? DATA1 : highreg_output; // mux

// registers
always @ (posedge CLK or posedge RST)
begin
        if (RST) begin
                highreg_output = 0;
                lowreg_output = 0;
        end else begin
                        if (LDPRE)
                        highreg_output = DATA2;
                        if (LDCOMP)
                        lowreg_output = DATA2;
        end
end

// counter
always @ (posedge CLK or posedge RST)
begin
        if (RST)
                DATA0 = 0
        else if (compare_output) // load
                DATA0 = mux_output;
        else
                DATA0 = DATA0 + 1;
end

endmodule
```

705

701

FIG. 8A

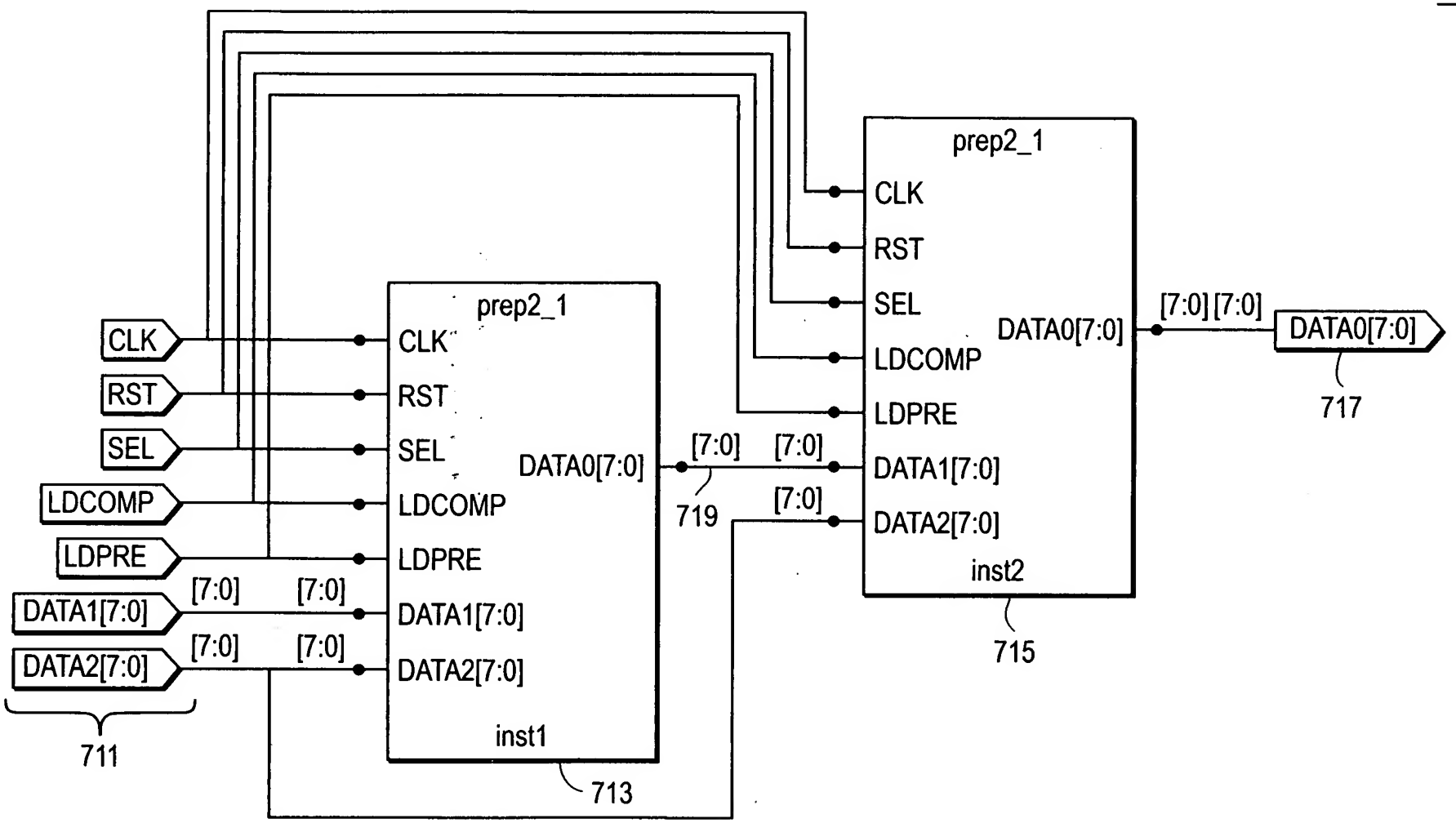
prep2_1
CLK
RST
SEL
LDCOMP
LDPRE
DATA1[7:0]
DATA2[7:0]
DATA0[7:0]
inst2
715
[7:0] [7:0]
DATA0[7:0]
717
[7:0]
[7:0]
719
[7:0]
prep2_1
CLK
RST
SEL
LDCOMP
LDPRE
DATA1[7:0]
DATA2[7:0]
DATA0[7:0]
inst1
713
CLK
RST
SEL
LDCOMP
LDPRE
[7:0]
[7:0]
DATA1[7:0]
[7:0]
[7:0]
DATA2[7:0]
711

FIG. 8B

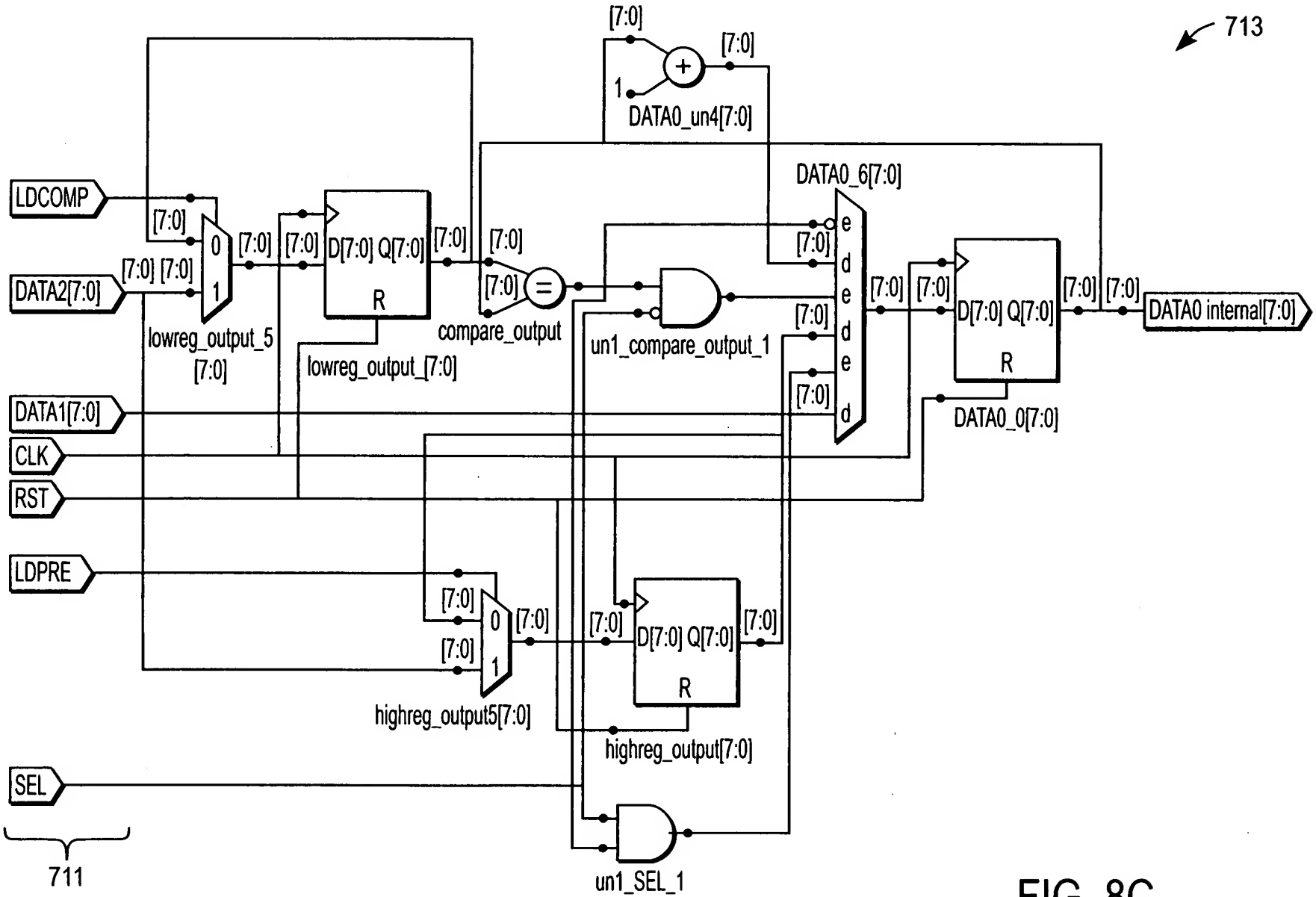
713
[7:0]
[7:0]
1
DATA0_un4[7:0]
LDCOMP
[7:0]
DATA2[7:0]
[7:0] [7:0]
0
1
lowreg_output_5
[7:0]
[7:0]
[7:0]
D[7:0] Q[7:0]
R
[7:0]
lowreg_output_[7:0]
[7:0]
[7:0]
=
compare_output
un1_compare_output_1
DATA0_6[7:0]
[7:0]
[7:0]
[7:0]
e
d
e
d
e
d
[7:0]
[7:0]
D[7:0] Q[7:0]
R
[7:0]
[7:0]
DATA0 internal[7:0]
DATA0_0[7:0]
DATA1[7:0]
CLK
RST
LDPRE
[7:0]
[7:0]
0
1
[7:0]
highreg_output5[7:0]
[7:0]
[7:0]
D[7:0] Q[7:0]
R
[7:0]
highreg_output[7:0]
SEL
711
un1_SEL_1

FIG. 8C

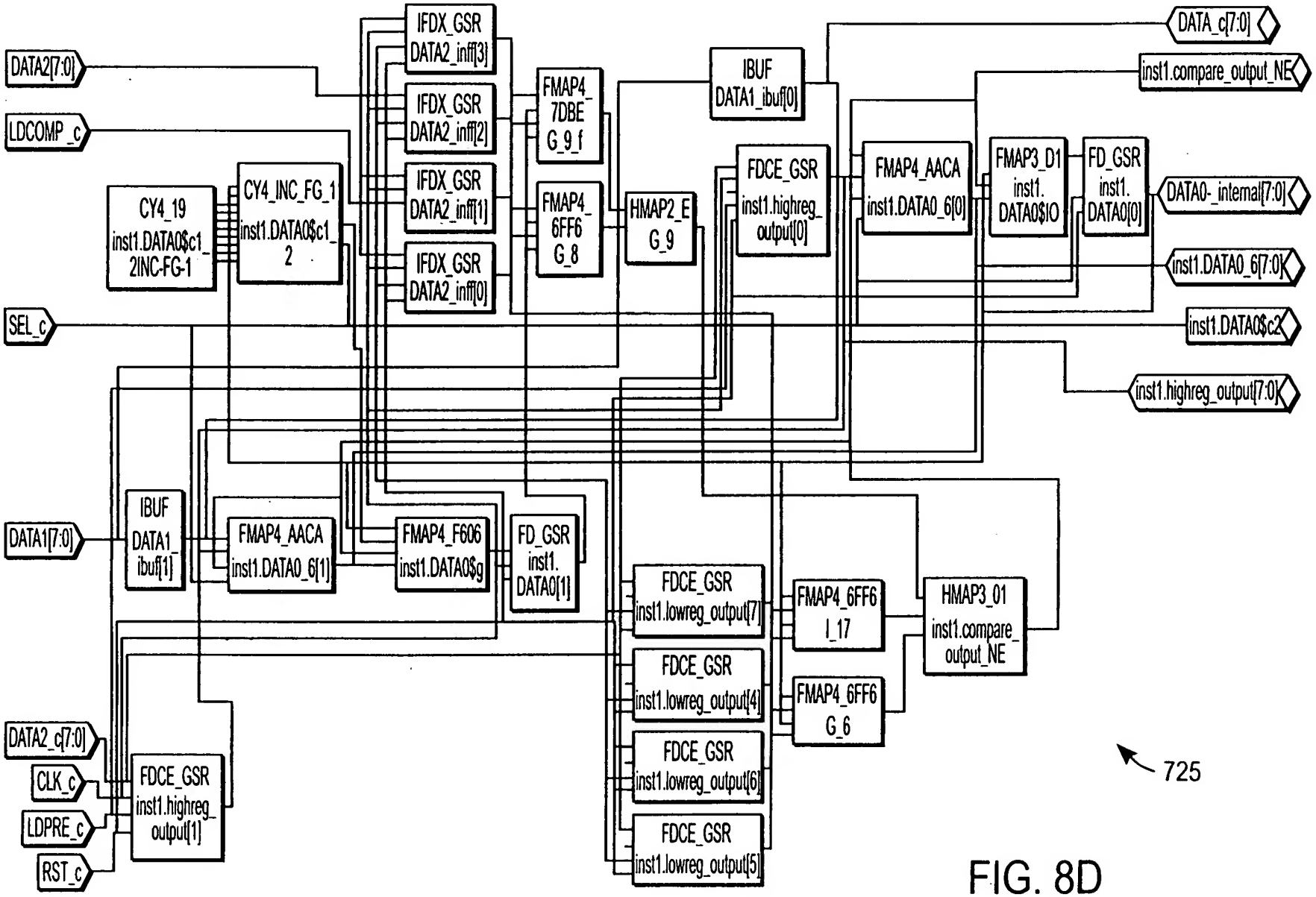
DATA2[7:0]
LDCOMP_c
SEL_c
DATA1[7:0]
DATA2_c[7:0]
CLK_c
LDPRE_c
RST_c
CY4_19
inst1.DATA0$c1_2INC-FG-1
CY4_INC_FG_1
inst1.DATA0$c1_2
IFDX_GSR
DATA2_inff[3]
IFDX_GSR
DATA2_inff[2]
IFDX_GSR
DATA2_inff[1]
IFDX_GSR
DATA2_inff[0]
FMAP4_7DBEG_9_f
FMAP4_6FF6G_8
HMAP2_EG_9
IBUF
DATA1_ibuf[0]
FDCE_GSR
inst1.highreg_output[0]
FMAP4_AACA
inst1.DATA0_6[0]
FMAP3_D1
inst1.DATA0$IO
FD_GSR
inst1.DATA0[0]
DATA_c[7:0]
inst1.compare_output_NE
DATA0-_internal[7:0]
inst1.DATA0_6[7:0]
inst1.DATA0$c2
inst1.highreg_output[7:0]
IBUF
DATA1_ibuf[1]
FMAP4_AACA
inst1.DATA0_6[1]
FMAP4_F606
inst1.DATA0$g
FD_GSR
inst1.DATA0[1]
FDCE_GSR
inst1.highreg_output[1]
FDCE_GSR
inst1.lowreg_output[7]
FDCE_GSR
inst1.lowreg_output[4]
FDCE_GSR
inst1.lowreg_output[6]
FDCE_GSR
inst1.lowreg_output[5]
FMAP4_6FF6I_17
FMAP4_6FF6G_6
HMAP3_01
inst1.compare_output_NE
725

FIG. 8D

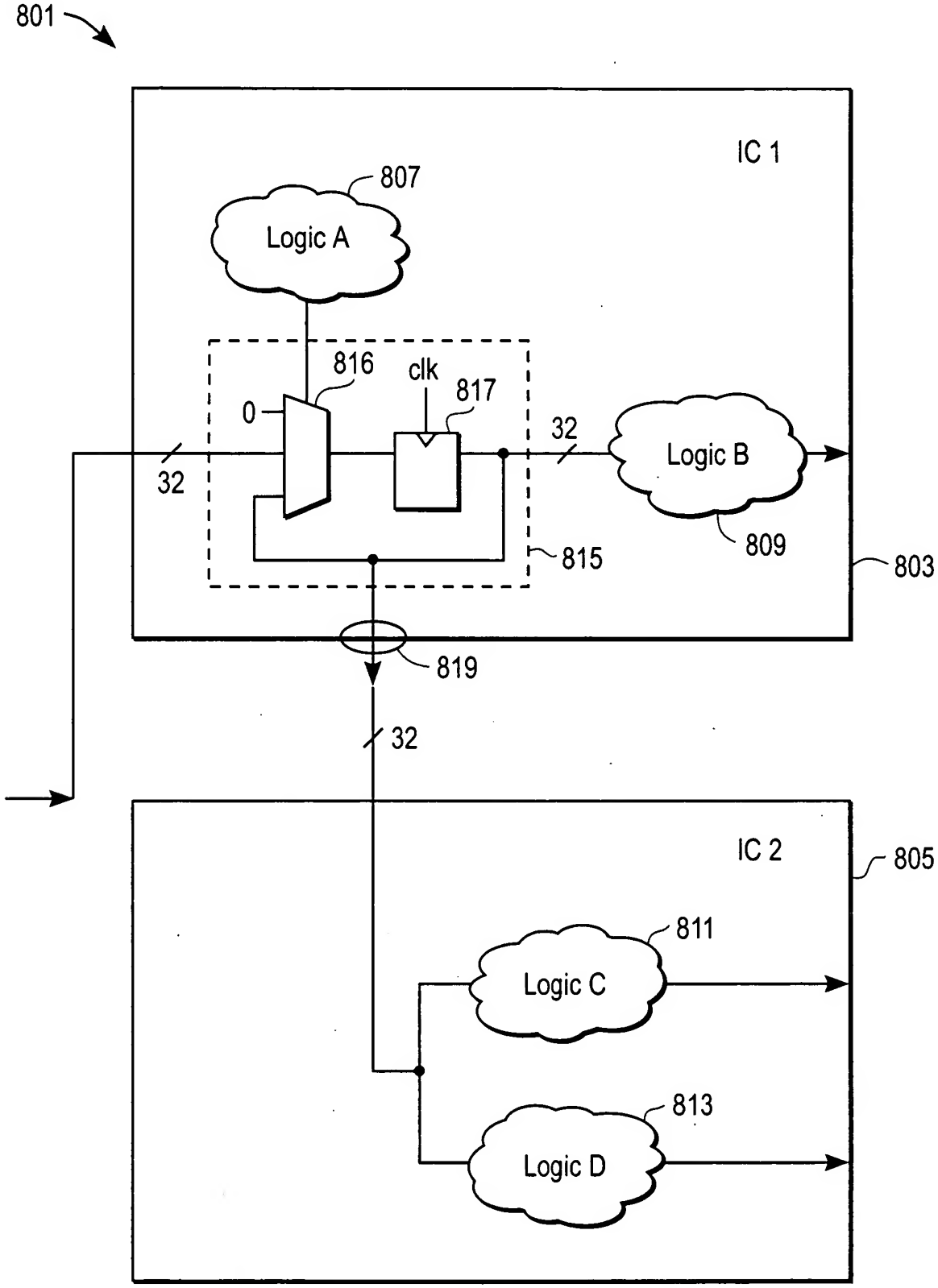
801
IC 1
807
Logic A
816
clk
817
0
32
32
Logic B
809
815
803
819
32
IC 2
805
811
Logic C
813
Logic D

FIG. 9A

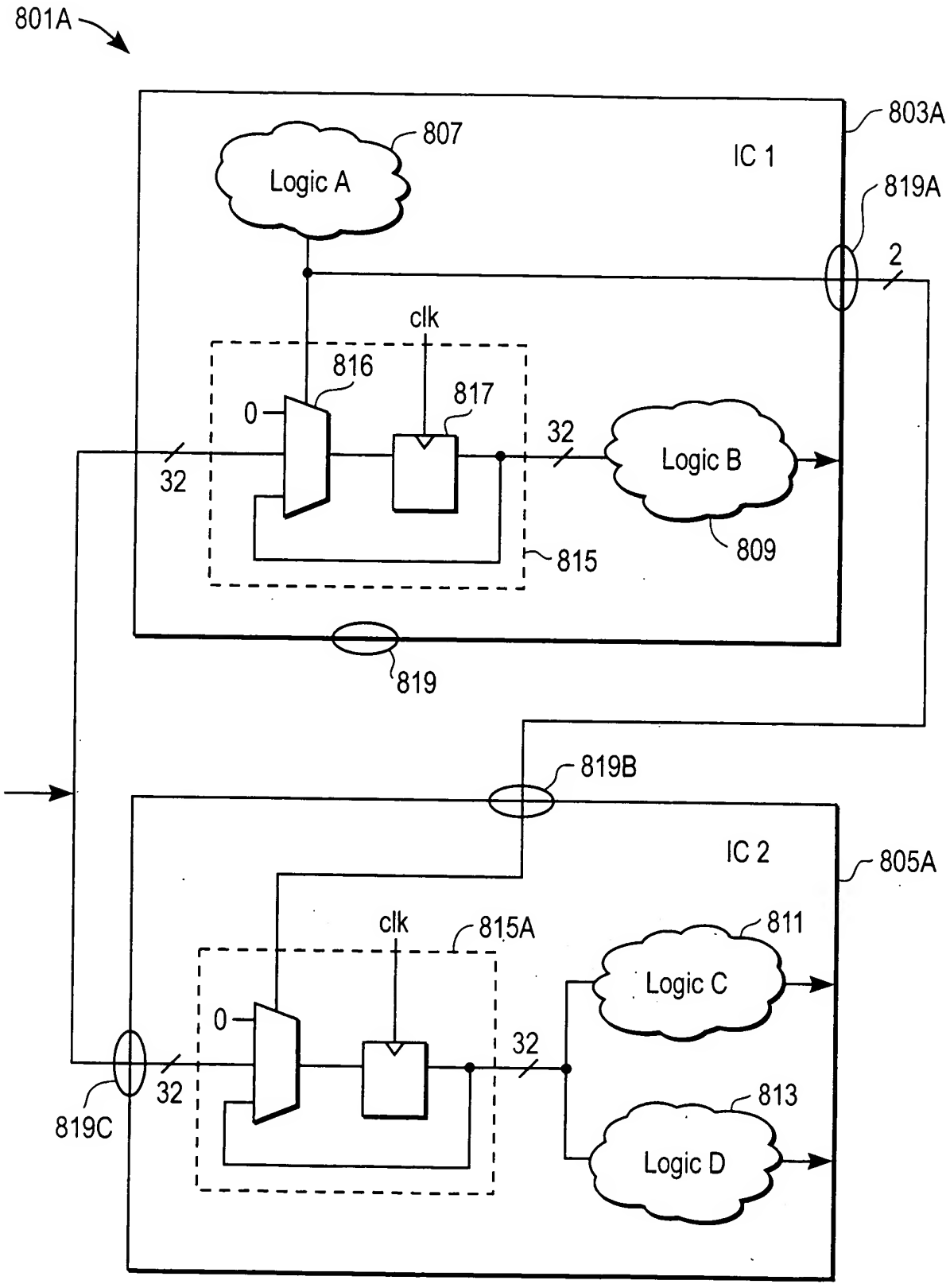
801A
803A
IC 1
807
Logic A
819A
2
clk
816
817
0
32
32
Logic B
809
815
819
819B
IC 2
805A
clk
815A
811
Logic C
0
32
32
819C
813
Logic D

FIG. 9B

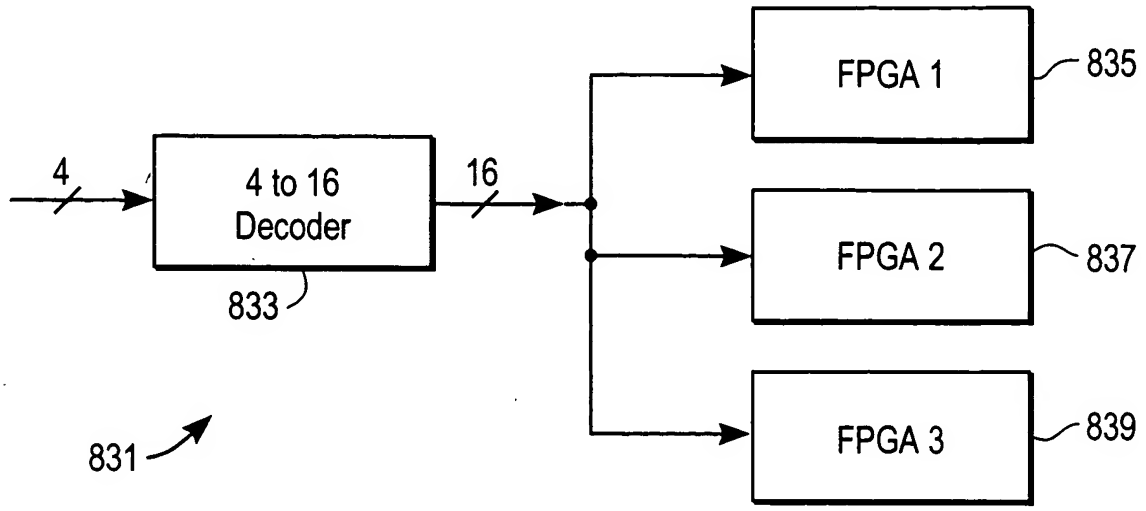
4
4 to 16
Decoder
16
833
831
FPGA 1
835
FPGA 2
837
FPGA 3
839

FIG. 9C

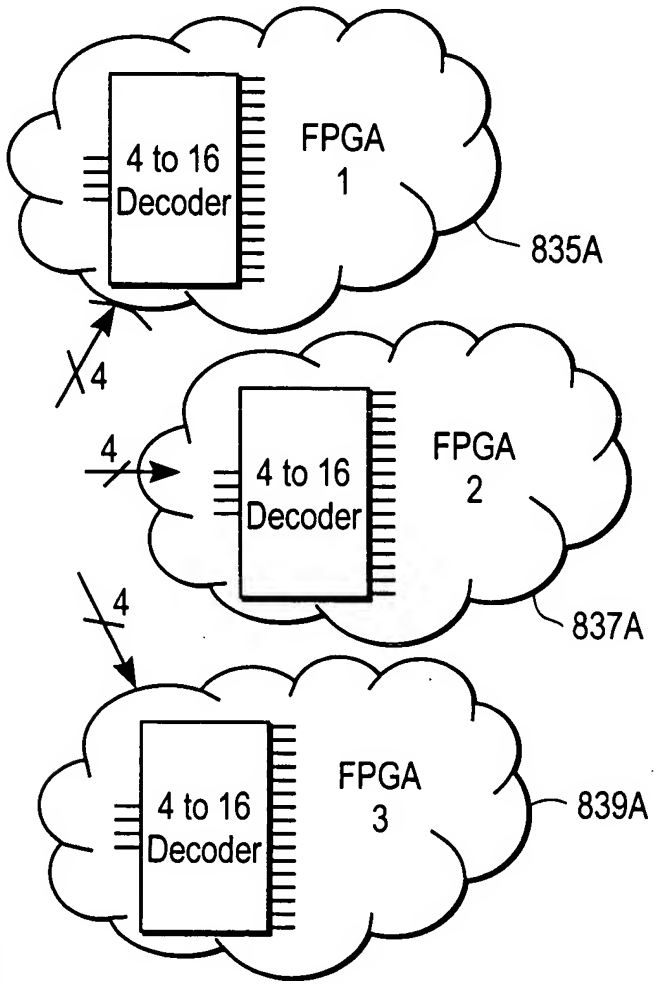
4 to 16
Decoder
FPGA
1
835A
4
4
4 to 16
Decoder
FPGA
2
837A
4
4 to 16
Decoder
FPGA
3
839A

FIG. 9D

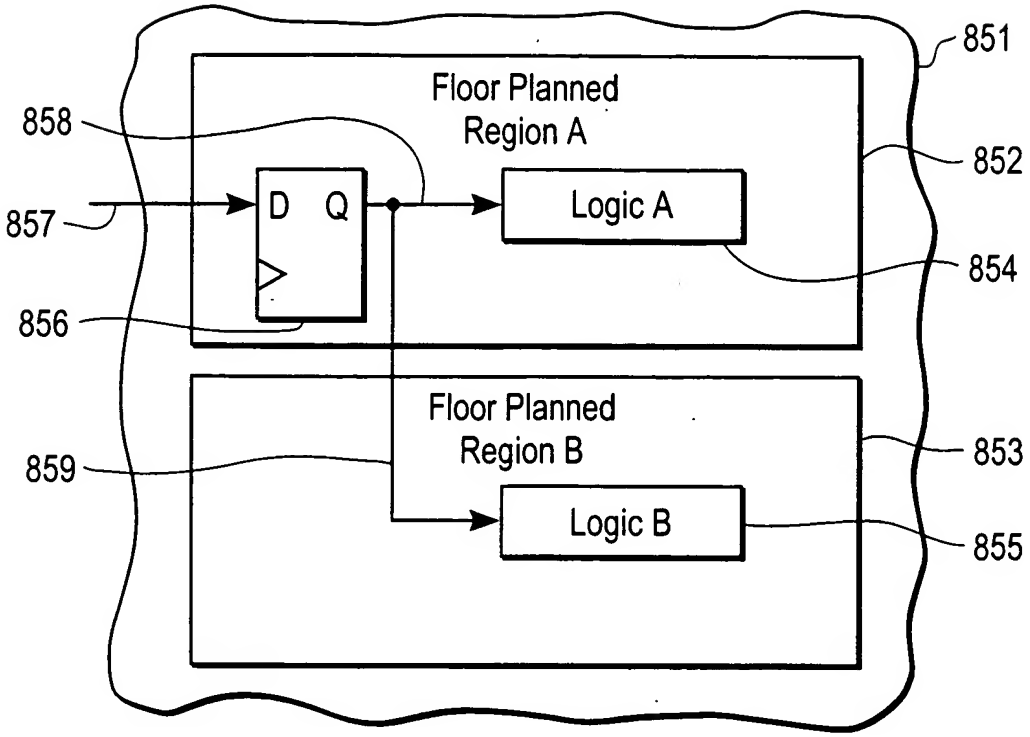
851
Floor Planned
Region A
858
852
857
D Q
Logic A
854
856
Floor Planned
Region B
853
859
Logic B
855

FIG. 9E

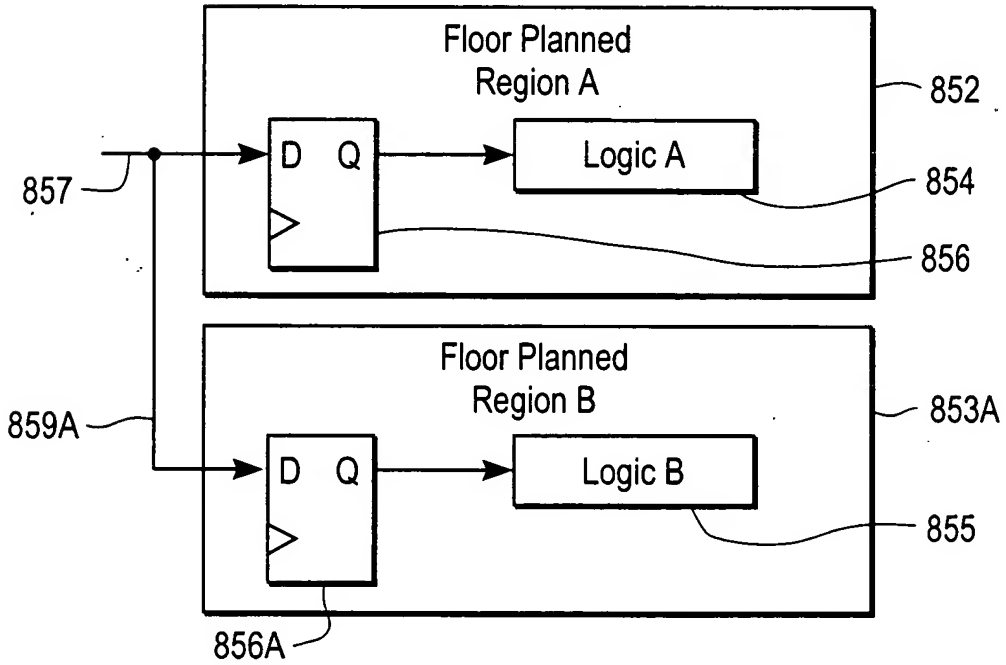
Floor Planned
Region A
852
D Q
Logic A
857
854
856
Floor Planned
Region B
853A
859A
D Q
Logic B
855
856A

FIG. 9F

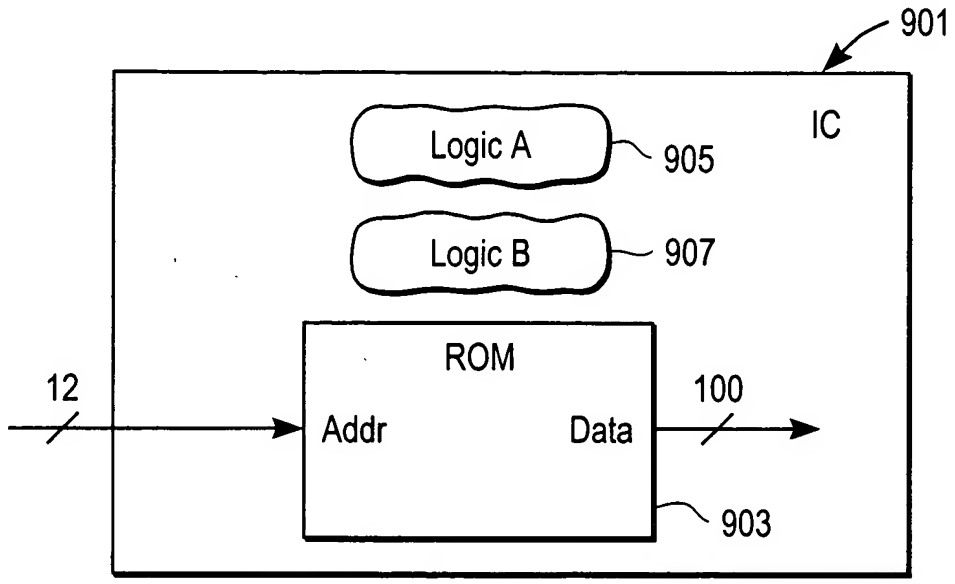

FIG. 10A

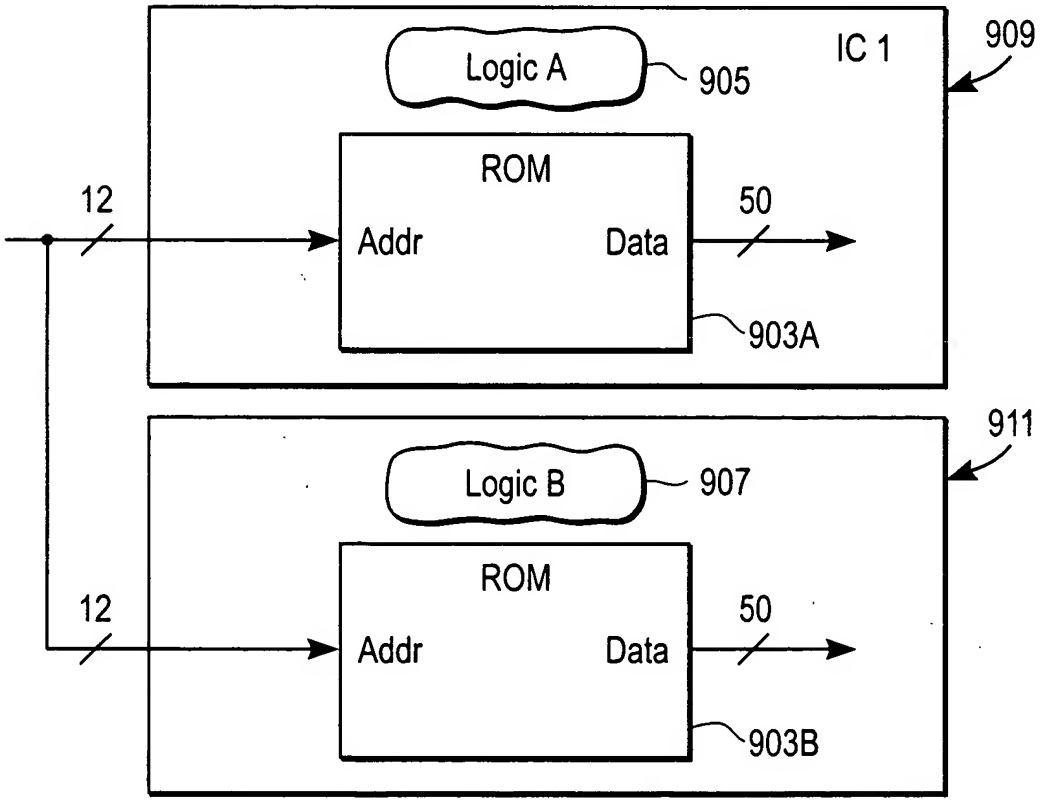

FIG. 10B

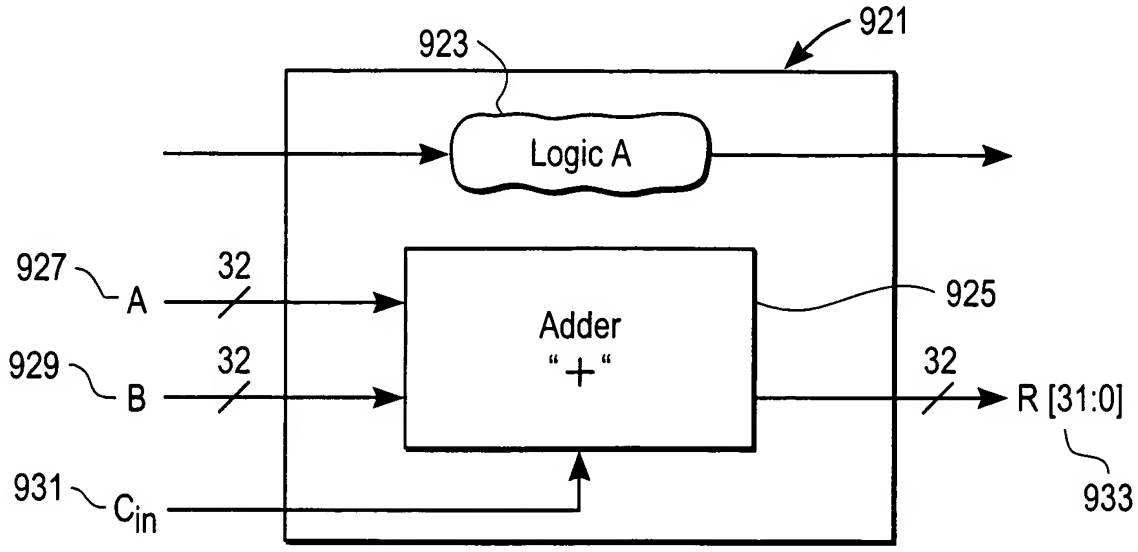

FIG. 10C

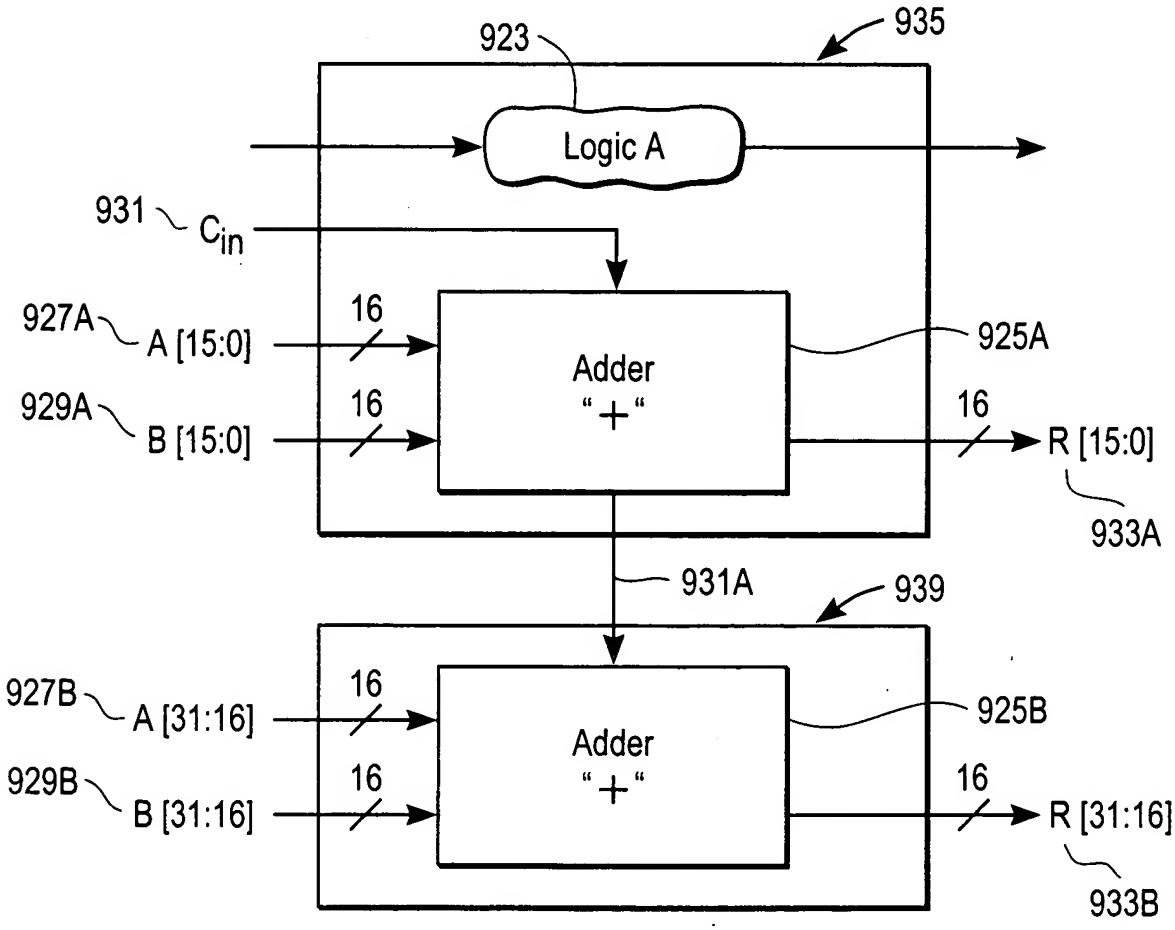

FIG. 10D

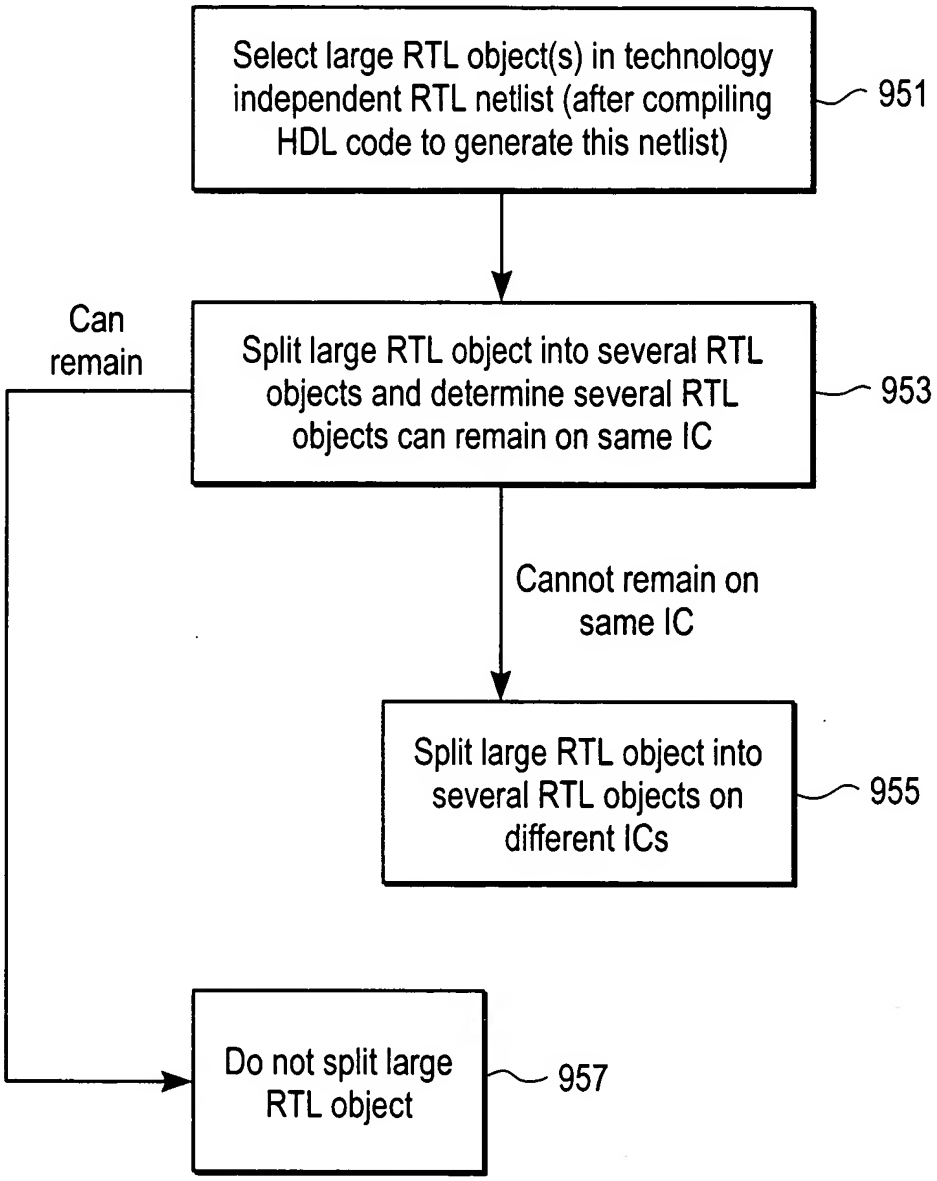
Select large RTL object(s) in technology independent RTL netlist (after compiling HDL code to generate this netlist)
951
Can remain
Split large RTL object into several RTL objects and determine several RTL objects can remain on same IC
953
Cannot remain on same IC
Split large RTL object into several RTL objects on different ICs
955
Do not split large RTL object
957

FIG. 10E

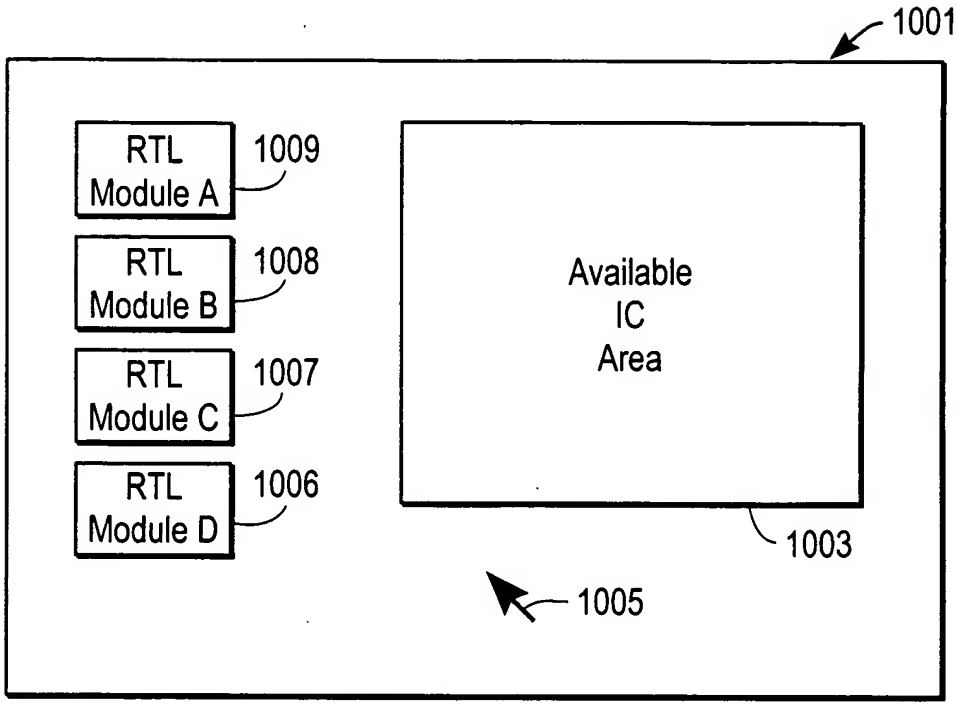

FIG. 11A

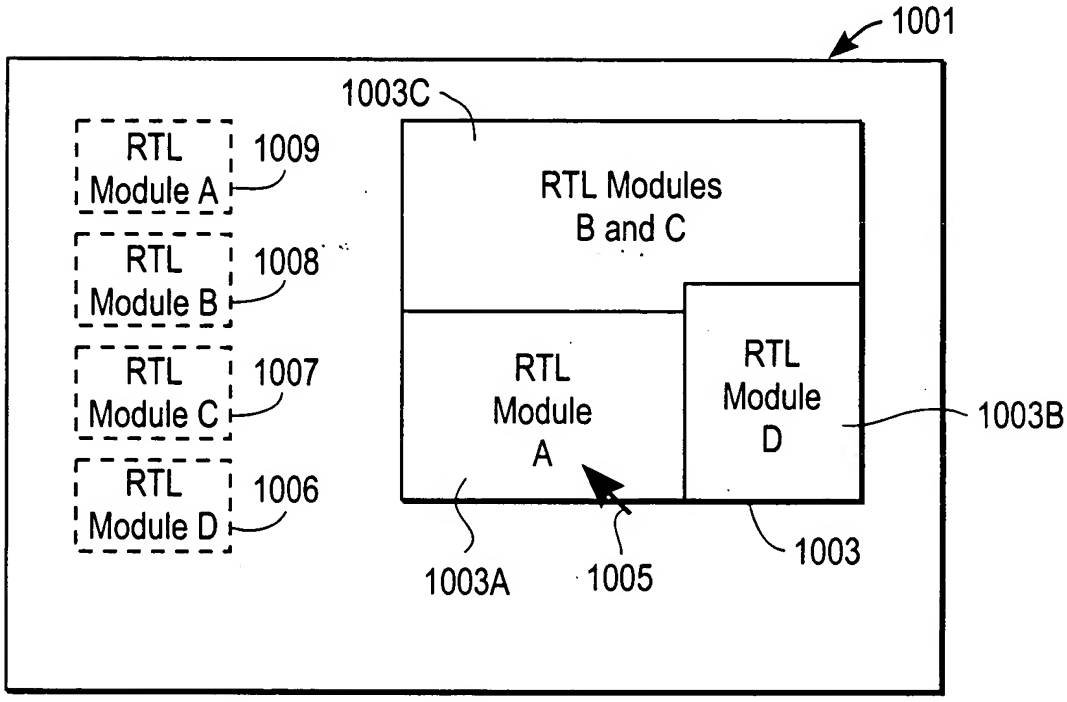

FIG. 11B

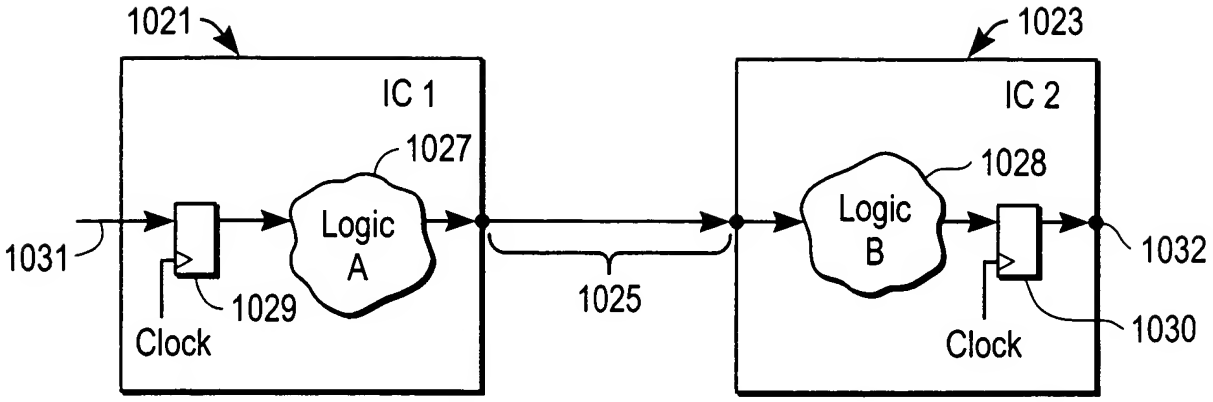
1021
IC 1
1027
Logic
A
1031
1029
Clock
1025
1023
IC 2
1028
Logic
B
1032
1030
Clock

FIG. 12

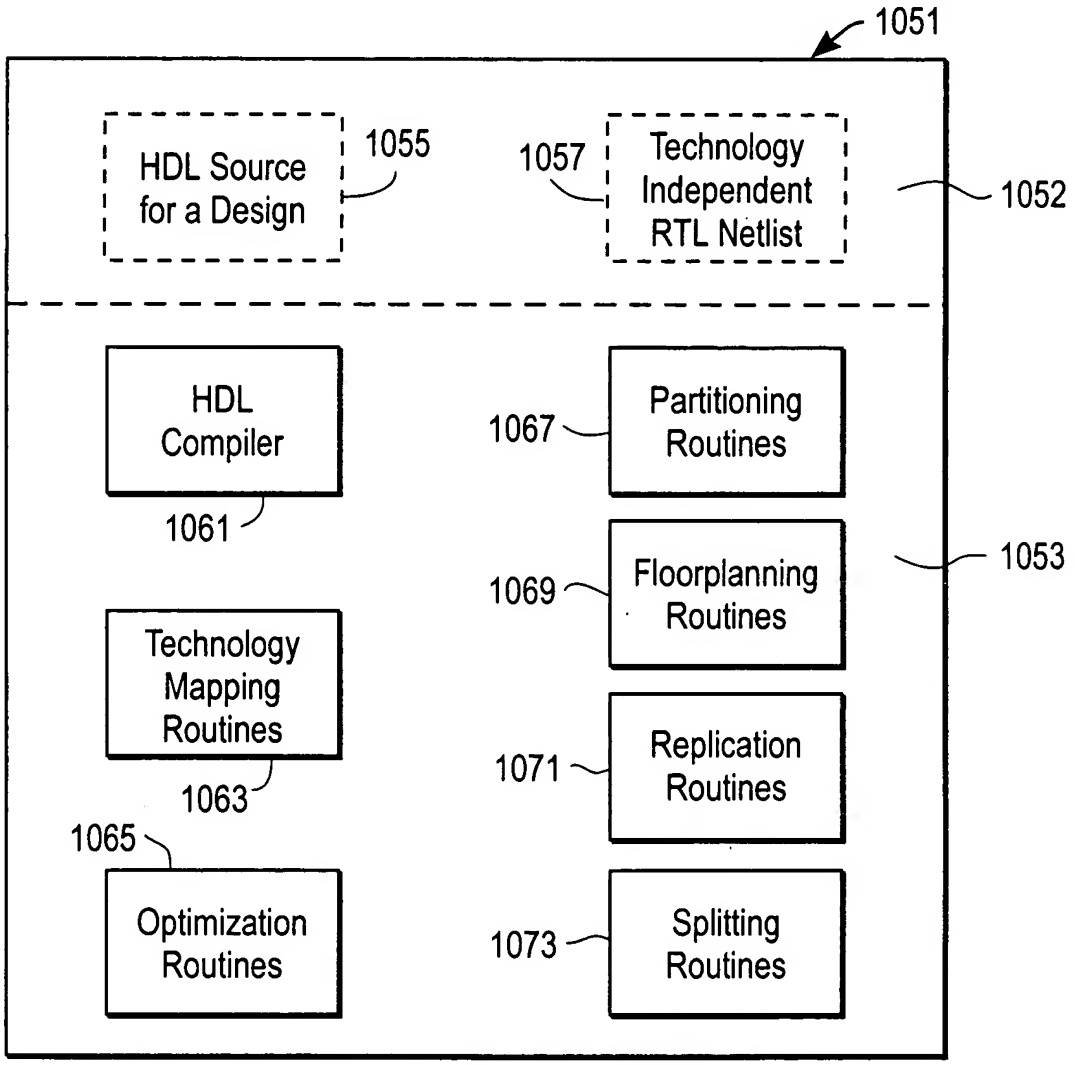
1051
HDL Source
for a Design
1055
1057
Technology
Independent
RTL Netlist
1052
HDL
Compiler
1061
Technology
Mapping
Routines
1063
1065
Optimization
Routines
1067
Partitioning
Routines
1069
Floorplanning
Routines
1053
1071
Replication
Routines
1073
Splitting
Routines

FIG. 13